Problem 1: Real-Time Table Filtering

Develop a dynamic table that displays a list of users. Include a search input field that allows users to filter table rows based on the user's name. The table should update in real-time as the user types in the search input.

HTML Structure:

```
<!DOCTYPE html>
<html>
<head>
  <title>Dynamic Table Filtering</title>
  <style>
    #filterInput {
      margin-bottom: 20px;
      padding: 8px;
      width: 200px;
    }
    table {
      width: 50%;
      border-collapse: collapse;
      margin-bottom: 20px;
    }
    th, td {
      padding: 12px;
      border: 1px solid #ddd;
    }
    tr.hidden {
      display: none;
    }
  </style>
</head>
<body>

<input type="text" id="filterInput" placeholder="Search for names...">
<table id="userTable">
```

```
  <thead>
    <tr>
      <th>Name</th>
      <th>Email</th>
    </tr>
  </thead>
  <tbody>
    <!-- Rows will be dynamically generated -->
  </tbody>
</table>

<script src="script.js"></script>
</body>
</html>
```

Requirements:

1. **Dynamic Population:**
    - Use JavaScript to populate the table with an array of user objects containing `name` and `email`.
2. **Real-Time Filtering:**
    - Implement a filter function that hides table rows not matching the search query in real-time as the user types.
3. **Case-Insensitive Search:**
    - The search should be case-insensitive, matching any part of the user's name.

Solution:

JavaScript Code (`script.js`):

```
// Array of user objects
let users = [
  { name: 'Alice Johnson', email: 'alice@example.com' },
```

```
  { name: 'Bob Smith', email: 'bob@example.com' },
  { name: 'Charlie Davis', email: 'charlie@example.com' },
  { name: 'Diana Evans', email: 'diana@example.com' },
  { name: 'Ethan Williams', email: 'ethan@example.com' },
];

// Function to populate the table
function populateTable() {
  let tbody = document.querySelector('#userTable tbody');
  tbody.innerHTML = ''; // Clear existing rows

  users.forEach(function(user) {
    let row = document.createElement('tr');

    let nameCell = document.createElement('td');
    nameCell.textContent = user.name;

    let emailCell = document.createElement('td');
    emailCell.textContent = user.email;

    row.appendChild(nameCell);
    row.appendChild(emailCell);
    tbody.appendChild(row);
  });
}

// Function to filter the table
function filterTable() {
  let filterValue = document.getElementById('filterInput').value.toLowerCase();
  let rows = document.querySelectorAll('#userTable tbody tr');

  rows.forEach(function(row) {
    let nameCell = row.cells[0];
    if (nameCell) {
      let nameText = nameCell.textContent.toLowerCase();
      if (nameText.indexOf(filterValue) > -1) {
        row.classList.remove('hidden');
```

```
      } else {
        row.classList.add('hidden');
      }
    }
  });
}

// Event listener for the filter input
let filterInput = document.getElementById('filterInput');
filterInput.addEventListener('input', filterTable);

// Initial population of the table
populateTable();
```

Explanation:

- **Data Source:**
  - An array `users` contains user objects with `name` and `email` properties.
- **Table Population:**
  - The `populateTable` function iterates over the `users` array and creates table rows (`tr`) and cells (`td`) for each user.
  - The function clears any existing rows before populating to ensure the table is fresh.
- **Real-Time Filtering:**
  - The `filterTable` function is called every time the user types in the search input (`input` event).
  - It converts the search query and the name in each row to lowercase to make the search case-insensitive.
  - If the name includes the search query, the row remains visible; otherwise, it is hidden using the `hidden` CSS class.
- **Event Handling:**
  - An event listener is attached to the `filterInput` element to trigger the `filterTable` function on every input event.